Projeto 6.48

SOFTESTE - Software Brasileiro de Automação de Testes

Robert Pereira Pinto

Coordenação e Execução: FUMSOFT - Sociedade Mineira deSoftware

Objetivos e Justificativa

Desenvolvimento de uma tecnologia, em software livre, que promoverá a integração de ferramentas especializadas em testes e funcionará como uma arquitetura de automação do processo de execução de testes em software.

Atualmente, para cada tipo de teste existe um conjunto de ferramentas específicas. A arquitetura a ser desenvolvida possibilitará a comunicação entre diferentes ferramentas existentes, o que permitirá que o processo de testes seja realizado de forma prática, facilitando, assim, a sua gerência.

A motivação para este projeto surgiu da ausência de suporte para a integração dos diversos softwares livres voltados para as etapas do processo de teste.

Existem várias propostas de ferramentas que realizam os diferentes testes, porém a gerência do processo como um todo torna-se bastante onerosa, uma vez que essas ferramentas não possuem suporte para compartilhamento de informações. Além disso, esses softwares, geralmente, não geram relatórios que possam auxiliar as empresas a seguirem modelos de qualidade e produtividade de software como o CMMI (Capability Maturity Model Integrated), e ainda, modelos de processos de testes de software como o TMM (Test Maturity Model).

Metodologia

Em uma primeira etapa do projeto foram definidas as ferramentas para integrar a primeira versão da arquitetura Softeste. Esta foi subdividida em duas fases: Levantamento e seleção das melhores ferramentas; estudo das interfaces de comunicação das melhores ferramentas.

Os critérios utilizados para seleção das ferramentas foram os seguintes: possuir licença livre; portabilidade; manutenabilidade e documentação.

O desenvolvimento da arquitetura Softeste foi realizado utilizando a metodologia de desenvolvimento de software conhecida como PRAXIS.

Tecnologia Resultante

A arquitetura Softeste foi desenvolvida de forma a tornar o mais flexível possível a adoção e substituição de ferramentas agregadas ao processo de testes. Basicamente, foram implementados vários elementos que compartilham informações por meio de um núcleo (*Middleware*).

Para melhor entender a estrutura, algumas definições se fazem necessárias:

– ***Scripts***: São entidades do Softeste que realizam conjuntos de ações com uma determinada finalidade.

– **Repositórios**: São recursos utilizados para armazenamento dos sistemas a serem testados ou mesmo dos arquivos contendo códigos e definições de testes a serem realizados.

– **Linha de Base**: Refere-se a uma imagem do projeto de software em um determinado momento. Neste contexto, é utilizada para controlar qual é o estado em que o sistema se encontra durante a execução de uma determinada bateria de testes.

Na figura 1 é apresentado o desenho esquemático da arquitetura Softeste. É possível perceber que existem seis tipos básicos de elementos que trocam informações e realizam operações específicas:

– **Ferramentas de apoio**: utilizadas para auxiliar a criação e a execução dos testes, estas são responsáveis pela compilação, controle de versão de projeto e gerenciamento de relatórios.

– ***Drivers* de comunicação**: utilizados para facilitar a comunicação entre as diversas ferramentas associadas à arquitetura Softeste e o *middleware*. Estes possuem um conjunto de definições padrão que permite a troca transparente de informações entre as ferramentas externas e o *middleware*.

– **Escalonador**: responsável pelo agendamento das execuções de baterias de testes. Este gerenciador aciona o *middleware* sempre que uma nova tarefa deve ser executada.

– **Ferramentas de testes**: voltadas para a criação e execução de testes bem como a gerência dos erros encontrados. Como possuem funções especializadas, porém voltadas para o mesmo fim, estão associadas a um único e grande componente da arquitetura Softeste.

– **Banco de dados**: tem a função de armazenamento dos *scripts*, dos parâmetros gerais da arquitetura e das informações a serem utilizadas para a geração dos relatórios estratégicos.

– ***Middleware**:* é responsável por interceptar as informações geradas pelas ferramentas de testes, tratá-las e retransmití-las para as ferramentas de controle de erros. Este está definido com maiores detalhes na próxima seção.

Middleware

Conforme já mencionado, o *middleware* funciona como um núcleo do Softeste que controla todas as ações executadas pelo sistema. Ele é

responsável por coletar as informações produzidas pelas ferramentas associadas, processá-las e encaminhá-las às etapas subsequentes, uma vez que realiza a comunicação entre os diversos módulos do Softeste.

A sua composição principal envolve classes de *scripts* e classes de controle. A figura 2 apresenta o esquema de comunicação entre os subsistemas que compõem o *middleware*. Como é possível perceber existe um núcleo interno composto pelos sub-sistemas de Cópia de Código dos Repositórios, Geração de Linha de Base, Registro de Erros e Compilação.

Estes são básicos para o funcionamento do Softeste sendo acionados pelos demais sub-sistemas: Criação Automática de Testes, Execução de Testes e Análise de Código.

O sub-sistema Criação Automática de Testes gera testes que devem ser utilizados pelos sub-sistemas Análise de Código e Execução de Testes.

Dada a funcionalidade principal do Escalonador que é o disparo de baterias de testes, este deve ter acesso aos *scripts* gerados pelos subsistemas Análise de Código, Criação Automática de Testes e Execução de Testes.

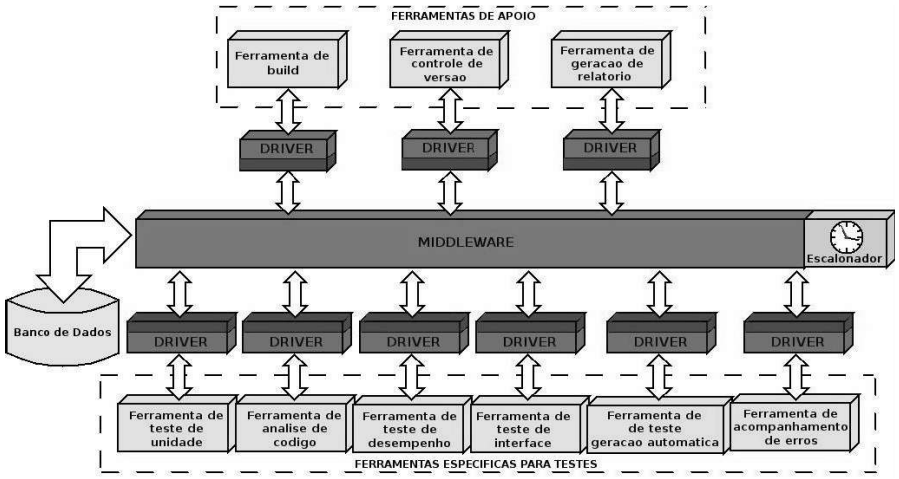

Figura 1: Arquitetura Softeste

Módulos do Sistema Softeste

A arquitetura Softeste foi subdividida em dez módulos principais que funcionam de forma cooperada. A maioria desses módulos são interfaces entre o usuário e os sub-sistemas implementados dentro do *middleware*.

Conheça a descrição de cada um dos módulos:

Configuração

Responsável pelo agrupamento das informações gerais utilizadas pelos diversos pontos da arquitetura. Neste módulo, o usuário administrador pode gerenciar informações como o idioma no qual as interfaces deverão se apresentar, o acesso de demais usuários e os sistemas externos reconhecidos pelo Softeste.

Cadastro de Sistemas de Cópia de Código do Repositório

Cadastra os sistemas responsáveis por buscar os códigos fontes a serem testados ou por armazenar os códigos e definições de testes gerados pelos usuários do Softeste. Este armazenamento é realizado por meio da utilização de sistemas de repositórios, também conhecidos como sistemas de controle de versão, que estejam cadastrados no módulo de configuração.

Cadastro de Sistemas de Registro de Erros

Captura os erros gerados durante o processo de teste realizado no Softeste e armazena-os nos sistemas de controle de erros informados pelos usuários da arquitetura.

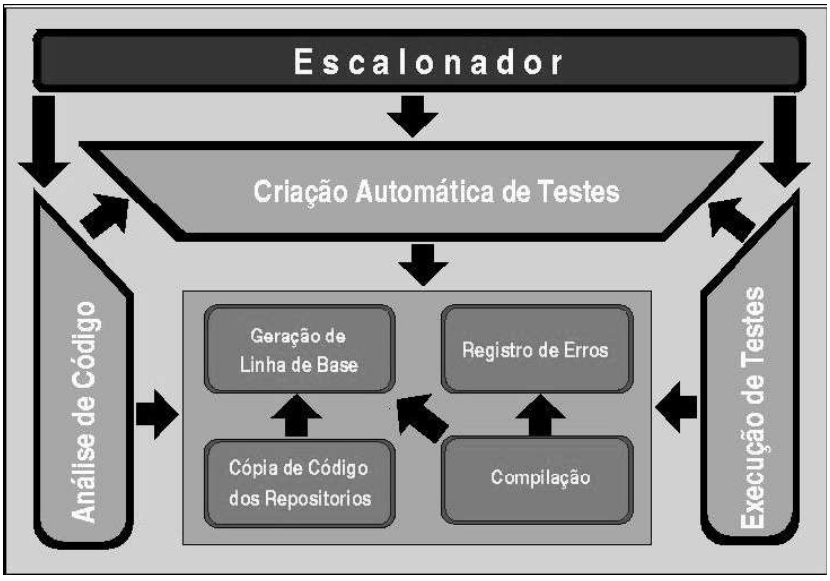

Figura 2: Estrutura Interna do Middleware

Cadastro de Sistemas Geradores de Linha de Base

Estes sistemas geram linhas de base do código a ser testado.

Cadastro de Sistemas Compiladores de Projeto

Compila linhas de base de aplicações a serem testadas. Quando requisitados isoladamente, esses se comunicam com algum sistema de geração de linha de base para que este gere uma nova linha a ser compilada.

Cadastro de Sistemas de Criação Automática de Testes

Um dos principais módulos da arquitetura Softeste responsável pelo cadastro de sistemas que, ao serem acionados, criam códigos fontes auxiliares que serão utilizados para a realização dos testes do sistema alvo. Desde que exista uma ferramenta apropriada cadastrada na arquitetura Softeste, este módulo permite a geração testes de análise de código, testes de unidade, testes de desempenho e testes de interface.

Cadastro de Sistemas Analisadores de Código

Este módulo foi implementado para a realização de análise de código de sistemas. Neste são cadastrados *scripts* que realizam tal operação. Os erros levantados automaticamente para uma determinada linha de base são repassados para *scripts* de registro de erros previamente cadastrados pelos usuários.

Cadastro de Sistemas de Execução Automática de Testes

Cria e dispara *scripts* que realizam testes de desempenho, interface ou de unidade. Esses são testes criados pelos usuários ou testes criados automaticamente pela própria arquitetura Softeste. Seu funcionamento é similar ao módulo analisador de código.

Escalonador de Testes

Agenda a criação e execução dos testes. Por meio deste é possível criar baterias e realizar testes de regressão nos sistemas. Os agendamentos podem ser periódicos e diversos testes podem ser acionados em um determinado momento.

Gerador de Relatórios

Permite que o gerente da instituição possa conhecer a produtividade de sua equipe de testes e de desenvolvimento, uma vez que os relatórios gerados fornecem informações como tempo médio para correção de erros, total de erros corrigidos em determinados períodos, produtividade por desenvolvedor, entre outras.

Estruturação do Servidor

A arquitetura Softeste está atualmente implementada como um servidor Web Java, estruturado nas camadas de apresentação, de negócios, persistência e de sistema. A figura 3 apresenta de forma esquemática como o servidor Softeste está estruturado.

Impacto e Contribuições Gerais

Por ser disponibilizado como um software livre, a arquitetura Softeste poderá ser distribuída de forma rápida pelas diversas empresas de desenvolvimento de software por todo o país, impactando diretamente na redução dos custos relacionados ao processo de testes.

Por se tratar de uma tecnologia independente de plataforma, o impacto em adoção de recursos computacionais é mínimo.

Uma empresa na qual já esteja sendo adotadas ferramentas específicas para determinados tipos de testes, pode facilmente adaptar o Softeste para que ele se comunique com essas, reduzindo desta forma os custos com implantações de novas ferramentas de testes.

Os investimentos em recursos de hardware não serão diretamente dependentes da adoção da arquitetura Softeste, e sim do processo de teste como um todo. Os principais fatores que podem motivar o investimento em novos recursos de hardwares são: plataforma de execução dos sistemas a serem testados; carga de teste a ser submetida aos sistemas; número de sistemas a serem testados; quantidade de testes

simultâneos que poderão ser realizados e ferramentas de testes legadas que possam ser associadas à arquitetura Softeste.

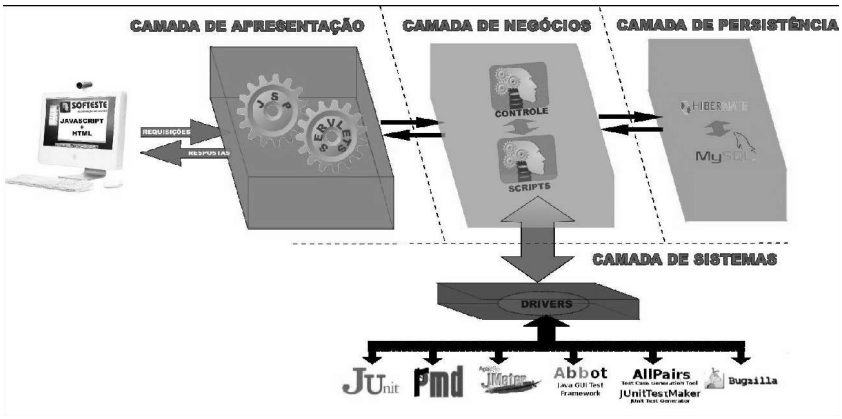

Figura 3: Infra-Estrutura do Servidor Softeste

Em se tratando das contribuições da tecnologia, o Softeste permitirá que pequenas e médias empresas de desenvolvimento de software possam automatizar seus processos de teste sem a necessidade de realizar altos investimentos.

Cabe citar ainda que, por ser disponibilizada sobre a licença LGPL (*Lesser General Public License*), qualquer organização que aderir à arquitetura terá o domínio total da tecnologia. Isto garante a independência de fornecedor, além de permitir que a própria empresa desenvolvedora de software realize evoluções na arquitetura Softeste, tornando-a mais adequada ao seu processo de teste.

A ampla divulgação e utilização do Softeste possibilitará a criação de um ambiente de automação de testes mais completo e adequado à realidade brasileira.

Desta forma, qualquer instituição, independente de tamanho, poderá ter acesso a uma tecnologia que permite a criação de software de maior qualidade e competitividade.

Características Inovadoras

O produto não recria nenhuma tecnologia já existente, ao mesmo tempo que também não se apresenta como algo absolutamente novo. No entanto, ao permitir a integração de ferramentas que até hoje trabalhavam isoladamente, a inovação surge do fato de que a nova tecnologia aprimora e renova o processo de testes de software, resultando em uma série de ganhos para a gerência do processo.

Desta forma, sistemas de testes bem difundidos no mercado podem ser utilizados em conjunto gerando informações que são compartilhadas por

meio do *middleware* do Softeste. É cabível lembrar que esses sistemas podem ser softwares livres ou produtos proprietários.

Conclusões e Perspectivas Futuras

Dado o cenário atual, pode-se concluir que esta arquitetura poderá atuar como um suporte para as empresas brasileiras produzirem software com maior qualidade e poderem, desta forma, posicionarem-se no mercado de maneira mais competitiva. Suporte como este é, em geral, fornecido somente por grandes ferramentas voltadas para desenvolvimento de sistemas, sendo essas proprietárias e que apresentam grandes custos de aquisição e implantação.

Uma vez que o Softeste é um software livre, apresentando-se altamente flexível com relação à adoção de diversas ferramentas específicas de testes, este possui um grande potencial de expansão, permitindo que seja adaptado à realidade das diversas empresas brasileiras de software, sejam elas micro, pequenas, médias ou grandes.

Atualmente, o projeto é disponibilizado em dois idiomas – português e inglês – e com suporte à internacionalização. Ao ser difundido para outros países, o sistema, consequentemente, absorverá novos recursos que servirão de apoio a todas as organizações clientes.

Existem diversos trabalhos a serem realizados, dentre eles podem ser destacados:

– criação de um mecanismo de geração automática dos casos de testes a partir dos casos de uso

– novos tipos de testes

– incorporação de mecanismos de sugestões de melhoria na configuração do ambiente de execução

Além desses, cabe ressaltar ainda a criação de uma comunidade de apoio ao desenvolvimento deste projeto. Deverá ser disponibilizado um portal, no qual usuários tenham condições de contribuir com o projeto, além de conhecer a proposta e poder adquirir fontes, instalador e documentação.

Referências

Sítio: "*Software QA Testing and Test Tool Resources*"

URL: http://www.aptest.com/resources.html

Sítio: "*GNU Lesser General Public License*"

URL: http://www.gnu.org/copyleft/lesser.html

Livro: Engenharia de software: fundamentos, métodos e padrões

WPP Filho - 2001 - Rio de Janeiro: Livros Técnicos e Científicos

Ferramenta: Abbot

URL: http://abbot.sourceforge.net

Ferramenta: Junit

URL: http://junit.org/index.html

Ferramenta: JunitTestMaker

URL: http://sourceforge.net/projects/junittestmaker/

Ferramenta: JunitTestMaker

URL: http://jakarta.apache.org/jmeter/

Artigo: *Developing A Testing Maturity Model For Software Test Process Evaluation And Improvement*

Ilene Burnstein, Taratip Suwanassart e Robert Carlson: Illinois Institute of Technology, Chicago